2005

LLuna - Designing a Distributed Virtual Presence System

Heiner Wolf
bluehands GmbH, wolf@bluehands.de

Christine Stumpf
bluehands GmbH, cs@bluehands.de

LLuna - Designing a Distributed Virtual Presence System

Heiner Wolf
bluehands GmbH & Co.mmunication KG
wolf@bluehands.de

Christine Stumpf
bluehands GmbH & Co.mmunication KG
cs@bluehands.de

ABSTRACT

This paper describes a research project on virtual presence, which strives to populate the Web. The Web is extended by a new layer that shows users on Web pages while they are present at virtual locations. The document first describes the requirements for a large-scale distributed system that fits to the Web. The requirements lead to design decisions, which build upon a publicly available messaging infrastructure. The paper presents a feature rich implementation and describes our experience with real users. It finally shows what virtual presence means for virtual communities.

Keywords

Virtual presence, web awareness, online community, social network, Jabber, MMORPG.

VIRTUAL PRESENCE

The Web is not just a collection of linked documents. The Web is a virtual space of millions of virtual places. Each Web site is a virtual place. Still, the Web is a rather primitive cyberspace compared to those "real" virtual worlds, which we know from Gibson [1], Stephenson [2], and other authors. As we surf the Web, we rather use the location metaphor than the document metaphor. Our impression is that we navigate to or go to a Web site rather than opening a hypertext document. And while we are there at a virtual location, there are other people virtually present at the same location at the same time. They are also walking around, jumping from page to page, and reading the same content. But we cannot see them. The Web consists of content and asynchronous communication. Presence, awareness of other people and synchronous communication are missing.

This gap is closed by the virtual presence concept. Virtual presence lets us see other people. It shows people who are at the same Web locations at the same time. Once we see them, we can communicate by chat, voice and video. Virtual presence is an enabling concept for synchronous interaction on the Web. Virtual presence transfers behavior from the real world to the virtual space. In the real world we are used to see people wherever we go. We see them on the street, in shops, and in public and private spaces. We usually ignore people and just pass by. But sometimes we communicate, if we meet a friend. In shops we ask the sales person and occasionally we even talk to strangers about a common topic. We want to enable a similar behavior in the virtual world. Actually, there is not much required. What we need is just being aware of each other and communication media. We therefore created Virtual Presence and let people talk.

Isolated Web Awareness

The Virtual Presence concept has been developed based on earlier concepts and systems. The idea of being aware of other users, who are reading the same page at the same time originated early in the history of the Web. First projects emerged in 1994. An important milestone was Virtual Places [3] where a meeting place was created for each data resources.

The CoBrow (Collaborative Browsing) [4] project introduced weighted awareness. Weighted awareness includes multiple parameters in the presence computation. Examples are the link distance between users in terms of the minimum number of hypertext references between their current Web pages, the duration of page visits, and the relation of the pages with respect to the content. CoBrow also created an individual presence for each user as opposed to systems, which created a single presence list for a data resource. Another Web awareness project, WebPlaces [5] applied the social proxy [6] display to communities of Web users.

These systems relied on Web server modifications or HTTP proxies for information gathering. They were small awareness islands. Clients were implemented in Java, which was the only way to show dynamic user interfaces in Web browsers.

Proprietary Systems

The second phase of Web awareness came during the Internet boom 1999. It was driven by startup companies, like Hypernix, NovaWiz, Cyland (product names: Gooey, Odigo, etc.). Their systems were centralized. All clients connected to the server network, which was operated by the providing company. They were using binary clients as user interface and for information gathering. The client showed a presence list of peers in a window, which was very similar to the static buddy list of IM clients.

These systems revealed their single point of failure, when venture capital dried out. Even though the failure had a financial reason, there was no infrastructure to keep the services running. Large players, which had joined the Web awareness family with co-browsing extensions to their IM products (ICQ Surf, T-Online Messenger), followed the market and stopped pilot projects.

Preparing for the Next Wave

It is not clear, why big providers like AOL and T-Online withdrew their Web awareness extensions. Critics would say that Web awareness is useless altogether. We cannot simply dismiss the argument. On the other hand, earlier systems had serious deficiencies. We believe that Web awareness is useful, if we can identify and avoid those deficiencies. We are trying to prove the theory by implementing a design, which improves on two aspects:

1. The operation of proprietary systems depended on a single provider. While the Web is hosted by millions of servers, Web awareness was hosted by a single (large) server. The service could not continue after the provider stopped supporting it. A distributed system for Web awareness could be more robust in this respect.
2. The presentation of peers as a presence list in the client window is not appropriate. The client window is separate from the browser window. There is no direct relation between the virtual location and the user names shown. Even if the client window was attached to the browser window, it contained a list of names. We are now using an avatar representation directly on the Web page, because movable and animated avatars illustrate the paradigm of virtual locations better than textual lists.

REQUIREMENTS AND BASIC DECISIONS

Disclaimer

Our basic assumption is, that people want to communicate in the virtual space. Many people might be comfortable with browsing privately without the need for communication on the Web. But, we see lots of asynchronous communication tools, like forums and some presence awareness like user online counters in forums. We assume, that people want to communicate in the virtual world, once they know, that it is possible.

Scalability

The potential user base for our virtual presence (VP) system is in the order of the number of Web users. Of course, we do not expect so many users any time soon. But we want to make sure that the system does not break down if it has 1 million users.

We identify two basic services: a virtual presence service and a communication service. The virtual presence service will make people aware of each other and the communication service handles basic chat communication and initiates advanced communication like voice. We plan for millions of connected clients and a similar number of conversations.

Such large numbers can be handled either by a very strong provider, like large instant message (IM) providers (AOL, MSN), or by a distributed network of federated servers. Since we are lacking a strong financial backing, we choose a distributed architecture. This suits our general conviction, that large systems should be distributed. More importantly: a distributed architecture fits well to the distributed architecture of the Web. We also try to avoid by design that anyone gets a single point of control.

The load will be distributed over multiple VP servers, which constitute the VP network. The question is how to distribute the load. Two strategies come to mind: We can either partition the number of users so, that a VP server handles a fraction of all users. Or we can partition the virtual space, so, that a VP server is responsible for a segment of the Web. Actually we do both. We have to distribute the client connections, because there are so many of them. We also partition the virtual space so that a VP server is only responsible for a part of the URL space.

Location Information

If we want to make people aware of each other, then we need to tell each of them about the presence of other users. The VP network uses information about the virtual location (URLs) of users. It matches the information and computes a presence list for each user, which basically contains IDs of other users. Then it forwards the lists to the VP client of the user. The client may then present the list to the user in whatever fashion is suitable.

So, we get many VP clients which talk to a distributed system of multiple VP servers. Each client telling its virtual location to a server of the VP network. This server must get information from all relevant clients. This means, that all clients on the same location must talk to the same server. Effectively, this VP server is responsible for a segment of the Web.

The next problem to solve is, that independent clients must agree on a VP server address for each virtual location without knowing each other. This can easily be solved in a centralized system where a single server is responsible for all locations. They would all talk to a single address. But we plan a distributed VP network. For a given virtual location (a URL) all clients must deduce the same VP server address in order to tell the server that they are there. The solution is, that they ask the Web server for the VP server address. On request by the VP client, the Web server returns the address of the VP server, which is responsible for virtual presence on the pages of the Web server. By providing the VP server address, the Web server actually controls the virtual presence on its pages.

Privacy

The VP system works with information about the URL trace of users while they are browsing the Web. This is a very serious subject. The VP network needs the information to work. On the other hand the user's privacy must be protected. We require, that there will no URLs of users be sent over the network. URL traces must not leave the user's computer. This sounds like a contradiction. But there is a solution.

Fortunately there is no need to send URLs over the network. What we really need is that all clients on the same URL agree on a common virtual location identifier. The identifier will not be the original URL. The solution is that VP clients create hashes of URLs with a message digest function like SHA1. Clients will compute the location identifier from a URL, but it will not be possible to regain the original URL from the identifier. VP clients map URLs to location identifiers and send only those over the network. There are more requirements to the mapping, but the important feature with respect to privacy is, that is uses a unidirectional hash function to hide the user's URLs from observers.

There remains the fact that people are visible to others on the same page. In the real world, we are used to being seen. At the same time, we dislike camera observation of public places. The same is true for virtual presence. We do not allow for observation of public virtual places. But we reveal our presence to other users who are virtually present. This is the very idea of virtual presence. It is a weaker privacy than without virtual presence. The level of privacy is closer to the one in the real world.

Presentation

A textual list of user IDs does not suit our needs. Our goal is, that users show up directly on the Web page as avatars. Users shall be able to move their avatars and virtually approach other people to indicate, that they want to chat.

Figure 2. A graphical user interface with avatars

We don't use a chat window. We let users chat in balloons. Everyone writes into the own chat bubble. An optional chat window provides a history of chat lines. The chat conversation is very much like a chat channel in ICQ, IRC or like Web based chat. But the user interface is adapted to the location metaphor.

Preliminary Architecture

A user enters Web pages with a favorite browser (1). The VP client, which may well be built into the browser, notices and requests the address of the responsible VP server (2). The VP client hashes the URL to get the location and tells the VP server that the user enters the location (3). The VP server gets similar requests from many clients and sends back a presence list (4). VP clients and VP servers communicate via a VP transport protocol that will be chosen later.

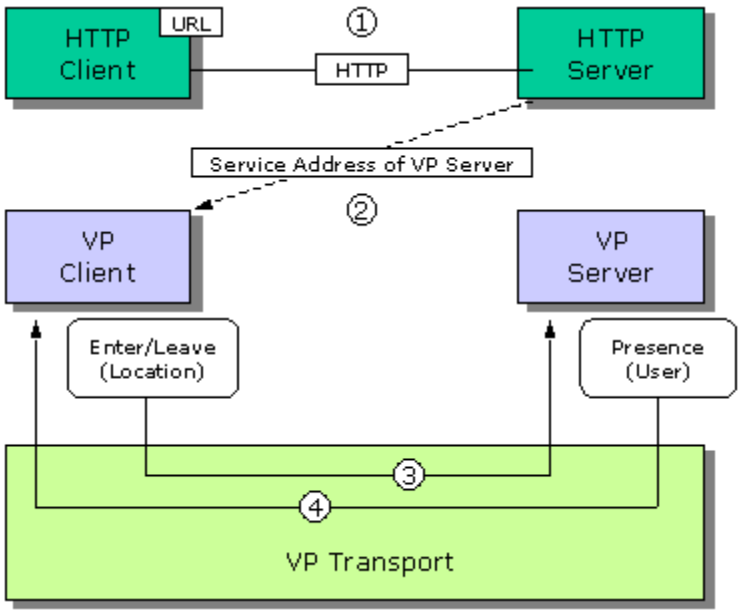

Figure 1. Virtual presence system architecture

Flexible Location Mapping

Flexible Location Mapping is a very technical requirement. URLs that users browse will be converted to location identifiers. A location ID, in turn, points to a VP server and this server connects the VP clients. The process of mapping URLs to location IDs controls where clients communicate. If you control the mapping process, then you control awareness and communication of users.

The design goals for the mapping are as follows:

- Allow operators of Web sites to control the mapping for the URL-space they control,
- Let Web sites opt out of virtual presence,
- Allow for hierarchical configuration for file system path based URLs,
- Support delegation,
- Allow for virtual presence without the cooperation of the Web site,
- Support commercial virtual presence servers and rented rooms,
- Be extensible to other protocols as virtual presence transport,
- Be easily implemented by Web site operators,
- Allow for distribution of the load of the default configuration server,
- Limit the privacy issues associated with virtual presence.

The requirements lead to a hierarchical mapping design that uses regular expressions and hashes, details omitted here. The most important features are, that it enables commercial virtual presence services and that Web site operators control the virtual presence on their pages. They can run their own VP servers and even moderate the communication on their pages.

Using Available Infrastructure

After experience with previous virtual presence projects, we chose to re-use an existing server infrastructure and protocol. Our goal was to implement only the client and use external server software. This decision limits what can be done in the protocol, because the client must comply with the protocol, but if offers several advantages. First of all the project can concentrate on the implementation of a client. More important in the medium term is, that server operators are much more likely to run servers with a wide range of support, than a server we would implement. Existing systems provide operation experience, documentation and an active community, if we choose the right one.

Fortunately there do exist server networks, which provide all we need. Almost any distributed chat network is suitable as a VP network:

- Chat channels would be used as locations and channel names become locations identifiers.
- The participant list of a chat channel serves as the virtual presence list.
- All chat systems offer a native chat capability.

Other desirable features are:

- A distributed per user storage facility for arbitrary data to store avatars and more.
- Extensibility to add virtual presence related features.

We concentrate on large chat networks with active open source communities. From these we choose the Jabber network (XMPP) [7] over IRC, because it provides additional services, like server storage. Jabber is an XML based instant message and presence network. The Jabber community developed a variety of protocols, clients, and servers. There are open source implementations and commercial ones of clients, servers, and libraries.

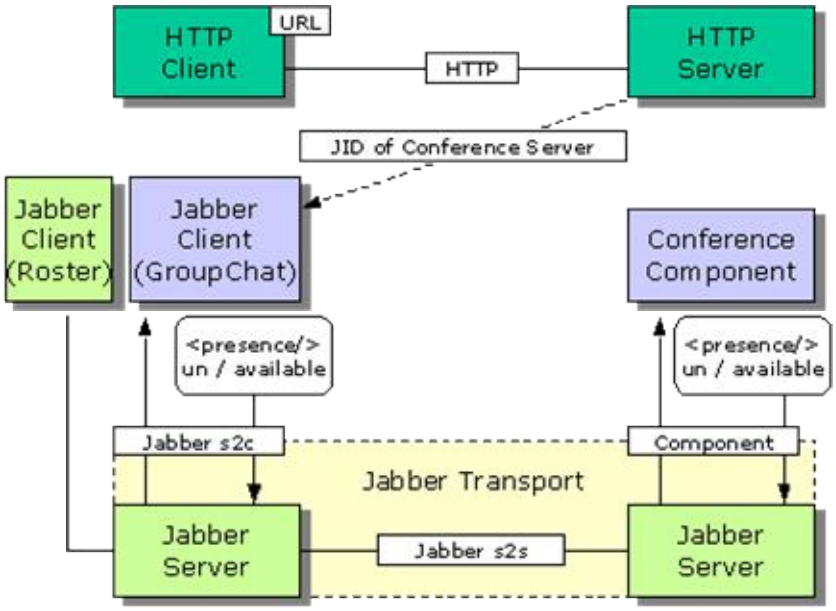

Figure 3. The Jabber network as transport and processing infrastructure

The Jabber network is structured so that a client connects to a server and servers interconnect with each other. Users and chat rooms are identified by so called Jabber IDs (JID). JIDs are similar to email addresses in that they identify a user at a server. Clients send messages to other clients via their home Jabber servers. In addition Jabber offers multi user chat, which can be

hosted by dedicated servers. Many servers share the load of client connections and other servers share the virtual presence load and chat conversations.

With the decision to use Jabber as transport infrastructure, the VP network becomes an overlay network on the Jabber net. We profit from the active community. We use and contribute to the protocol development. And there are already thousands of Jabber servers deployed, which can be used for the VP network.

IMPLEMENTATION

A Client ...

Much of the consideration above happened actually before implementation started. We could draw from our experience with earlier virtual presence systems. From the technical point of view, we implemented a Jabber group chat client with a graphical user interface, which automatically enters and leaves Jabber chat rooms while the user is browsing the Web. From the user's point of view we developed a program, which runs in the background, and as soon as you go to a Web site, it shows your avatar on the page and the avatars of other users, who are at the same Web page at the same time.

In addition to the basic functionality of showing avatars and chat, it provides a small video for each user, if a camera is available. It supports private chat for one-to-one conversations. Avatars can move around. People can approach others by moving the avatar closer. Avatars can be animated figures, which walk left and right like small game characters. The client has a whiteboard, which people use to paint on and point at the Web page. It includes co-browsing functions to enable guided tours.

... for Consumers

LLuna is a feature rich implementation, not a bare reference implementation for virtual presence. It tries to hide details of the transport infrastructure (Jabber) from the user. Users can use their existing Jabber instant message accounts, if they have one. But most users do not have such an account registered with a Jabber server. For these users LLuna, automatically creates an account on a public Jabber server.

End users do not care what is required to run the software. They just want it to work. As a project we are grateful, if someone downloads and tries the software. We do not bother users with details, at least not at the beginning. The first start is very quick and easy. Later, users can set various options and configure the client to their needs.

Technical Details

LLuna is compatible with most Jabber servers, specifically with two open source implementations, jabberd1.4 and jabberd2. The client specializes in virtual presence features rather than IM functions. But it can run simultaneously with other jabber clients. People who meet each other can click on the avatar and add LLuna users to the persistent buddy list of the IM client.

The client is compatible with any JEP-0045 (MUC = multi user conference) compliant conference component [8] and with older "Groupchat"-components. Password protection and moderation features require a MUC component.

The client detects URLs of the pages shown in local Web browser windows. It attaches to the browser process. The exact mode of operation depends on the browser software. LLuna maps the URL to a Jabber chat room ID (JID) any time the URL changes. It leaves the previous room and joins the next room. Jabber chat rooms are created dynamically by the first participant. Rooms can be configured to enable password protection, moderation and other advanced features.

Of course, changing rooms is only required, if the JID associated with the previous URL is different from the JID of the current URL. A typical user's Web browsing behavior includes multiple pages of the same DNS domain. We set the initial URL-JID mapping to create one room per domain (the mapping can be customized by the Web site). Combining both means, that there are fewer room changes than page changes. A page change requires the data transfer of several 10 KB up to few hundred KB with an average at 50-70 KB. This is a rough number based on a quick test of public news sites. A Jabber room change requires few hundred bytes. This means, that the relation between overall Web traffic and virtual presence traffic is in the order of 100/1.

Users can choose whether their own avatar is always visible or only if they meet other people. It turned out, that most users like to see their own avatar, even if they are alone on a Web page. Avatars occupy only a small fraction of the page. If an avatar hides content, then the user scrolls the page up or just moves it out of the way. If users do not want to be visible on certain pages, then they can easily opt out. They can also stop virtual presence by pressing an emergency button in the system's task bar.

Figure 4. Space hidden by avatars compared to the overall browser space

OPERATION EXPERIENCE

Strong Architecture

The LLuna VP client is available since 18 months. More than 20,000 users downloaded and installed LLuna. Initially the project operated a Jabber server as virtual presence and chat server. When the traffic and the number of client connection increased, the Jabber Software Foundation (JSF) offered to host the traffic. We could route the traffic to the JSF within minutes.

Since most users do not care about details, they are registered at a home Jabber server we propose. All the communication of a client runs over the home server. This includes all chat communication. Therefore, we only propose trusted Jabber servers as home-server for the inexperienced user. The distributed architecture allows us to choose trusted public servers, like the German CCC and the JSF and balance the load between them.

Another aspect of the system proved to be important. There are Websites, which operate their own chat server for LLuna, because they want to moderate the communication on their Web. They put up a configuration file on their Web server that tells all VP clients to use a specific chat server instead of the default server. Hence, they control the chat server. They can assign moderators with special privileges to ban, mute, or even kick users, if necessary.

When we started, there was no moderated chat in the Jabber world. We created a moderated chat component. While we were implementing the client, other Jabber developers came up with a multi user moderated chat component including a protocol specification. We now use this component, without diverting more resources from the client development.

Chicken and Egg

There is still a chicken and egg problem. Presence awareness needs users. People only benefit from being virtually present, if there are other people they can meet. 20,000 users over 18 months mean about 50 new users each day. Some meet on the project home page (http://www.lluna.de/). Others meet on their community site. But otherwise they barely meet anyone on the Web, because users are distributed over millions of domains and pages. Many early users stopped using LLuna after some time, because they did not meet anyone by chance.

There can be many reasons why a system is not being used. We can rule out that the software is bad. Most new users like the software. They tell this in conversations with project members when they first try it. Reactions reach from a "good, but not enough users" to "incredibly innovative" with a mean at a "cool" rating. Apparently the main problem is, that the critical mass of online users is missing. This is where communities come into play.

Communities for Virtual Presence

Communities can provide the critical mass for virtual presence in general and for LLuna. Not only because of the number of users. Members of a community often share a passion, hobby, profession, or similar living conditions. It is the common feature that makes them a community. The same feature lets community members focus on a subset of the Web. This increases the chance that people meet while they are browsing. Thus, it helps to solve the chicken and egg problem of virtual presence. But what can virtual presence do for communities?

VIRTUAL PRESENCE FOR COMMUNITIES

Currently, online communities build around Web sites. Members meet on the Web site indirectly and communicate asynchronously. Some also have chat channels. Virtual presence has two offers for communities:

- Community members can meet and recognize each other as members of a peer group everywhere in the virtual world. This is a new feature for the community.
- The community can extend its reach beyond the community Web site. It can bind members or even acquire new members, because the members can carry the community's message beyond the Web site.

The operator of a community site may not have the same goals as the community members. Many operators have a commercial interest in visits to the Web site, because they finance the site by banners, or partnerships with commercial services.

Ubiquitous awareness and interaction offered by virtual presence sometimes is contrary to business models, which are based on limitations. Examples are dating communities, where members pay for communication. We are sure that artificial limitations could be introduced into virtual presence to support these models, but they are not our focus. We concentrate on communities, where the operators can clearly benefit and show two examples:

- Social networks and
- Game communities.

Social Networks

Social networks (SN) store, evaluate, and display relationships between members [9]. The FOAF project is a good example [10] for a SN technology. Social networks are based on the idea, that even if I do not know a member, there is usually an indirect relation over other members. People can learn about their indirect relationships. If they search for a person, they get the partial graph, which connects the two of them.

A combination of virtual presence and a social network would greatly increase the value of the information stored in the SN. My VP client can use the SN to display the relationship to any other member I meet. Instead of entering a name into the social network software, the VP client searches the SN database automatically. The VP serves as a name provider for SN searches. It can uncover new relations, because the names space provided by VP will be very different from what users type in and search for manually.

People who meet on a Web site have a common interest, that shows up in the content. This is a natural reason for communication. Knowing from an SN component if and how they are related reduces the communication barrier. The practical value for SN members is that they meet people on content of common interest and they know that they have friends in common. This is true for networks of friends, but also for business oriented SNs. For the social network, virtual presence can be a feature of a premium membership.

Online Game Communities

Most massively multi user online role-playing games (MMORPG) see a very steep increase of monthly paying customers after the release. Within 1-2 year they saturate, then slowly decline when the public attention moves to new releases [11].

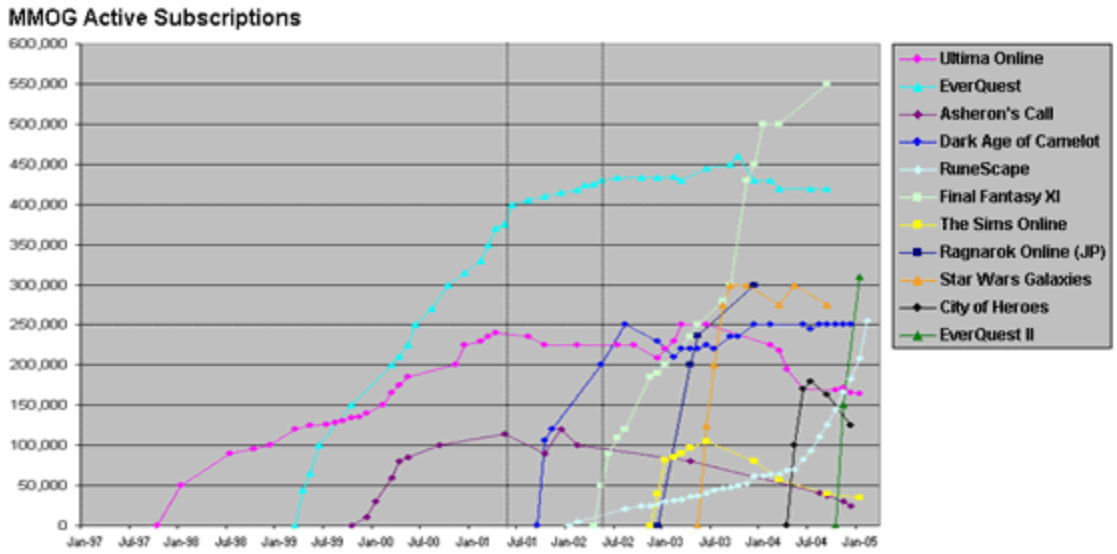

Figure 5. MMORPG active subscriptions over time [11]

Games rarely get new customers after one year. They try to keep their customer base up as long as possible by expansion packs, new features, and community support. Still, over time many users terminate the account and stop paying the monthly fee.

From the user's point of view, terminating the account is very painful. Users invest hundreds of hours over many months, even years into their game characters. They level it up and build social relationships. Users terminate their account only, if they are sure, that they will not have the time to continue playing. The decision is a difficult one and the reason to quit is that the character is not in use anymore.

Virtual presence gives a reason to keep an account open, because gamers can use their character beyond the game. They can walk on the Web with their character., Others can even see details and statistics of the character. They can continue on the Web, what they already do in-game. They walk around, being proud of their achievements, their reputation, or their items. And they recognize other players as members of their peer group easily. This is a useful feature for a game community in itself even when the community is new. But during the early phases games have enough features. A VP component is just one more feature.

After the peak, VP can become really important. VP gives a reason to keep accounts. A gamer who has no time to spend hours in the game, can still use the character outside. The character would not evolve anymore, but it would continue to be useful, even if there is no time to play. This may defer the decision to terminate the account. It can translate into months of continued payments of a significant portion of the would-be dropouts. This represents real money for the publisher. and an added value for the user.

SUMMARY AND PLANS

We have shown how we created a distributed virtual presence system. We derived important design decisions from requirements and implemented a system for end users. Our experience shows that users want to be aware of each other once they grasp the concept. But we also learned, that virtual presence needs a critical mass of active users.

The next steps are the integration of virtual presence and social networks, and cooperation with role-playing games. Our current client is already capable of displaying animated avatars. It would be only a one month effort in cooperation with a game publisher to allow in-game characters to walk on the Web. Future plans generalize even more. We regard the Web as one of many virtual world. Each online game spans a virtual world. We now allow for in-game characters to cross the border to the Web. They are just visitors on the Web. We could think of them visiting other virtual worlds as well. Game characters could cross the border to other virtual words as visitors. Virtual worlds could then specialize in a certain field. The Web is good for information retrieval, less for adventures. MMORPG are good for fighting, less for living. There are other more living oriented worlds, which provide impressive virtual housing. Our characters could live in one world, adventure in another, shop in the virtual mall and accompany us on the Web.

ACKNOWLEDGMENTS

This project has been funded by SURFnet bv and bluehands GmbH & Co.mmunication KG.

REFERENCES

1. Gibson, W. (1995) Neuromancer, Ace Books, Reissue edition May 1995, ISBN: 0441569595
2. Stephenson, N. (2000) Snow Crash, Bantam Doubleday Dell Pub., Reprint edition May 2, 2000, ISBN: 0553380958
3. Mass, Y. Virtual Places - Adding people to the web, Proceedings of the Third International World-Wide Web Conference 1995
4. Sidler, G., Scott, A. et al (1997) Collaborative Browsing in the World Wide Web, Proceedings of the 8th Joint European Networking Conference, Edinburgh, May 12.-15
5. Maglio, P.P., Barrett, R., WebPlaces: Adding People to the Web, Poster Proceedings of the Eighth International World Wide Web Conference 1999, <http://www.almaden.ibm.com/cs/wbi/papers/www8/wwwplaces-abstract.html>
6. Erickson, T., Smith, D. N., Kellogg, W. A., Laff, M., Richards, J. T., & Bradner, E. (1999). A sociotechnical approach to design: Social proxies, persistent conversations, and the design of Babble. Proceedings of Human Factors in Computing Systems, CHI '99. New York: ACM Press.
7. Adams, D J (2002) Programming Jabber, O'Reilly.
8. Saint-Andre, P (2005) JEP-0045: Multi-User Chat, Jabber Software Foundation, <http://www.jabber.org/jeps/jep-0045.html>
9. Hanneman, R. A. (2001) Introduction to Social Network Methods (PDF), <http://faculty.ucr.edu/~hanneman/SOC157/NETTEXT.PDF>
10. The FOAF project (2005) <http://www.foaf-project.org/>
11. Woodcock, Bruce Sterling (2005) An Analysis of MMOG Subscription Growth MMOGCHART.COM 12.0, <http://www.mmogchart.com>